ATTORNEY DOCKET No.
SUNMP348

PATENT APPLICATION

PROCESSOR CORE CLOCK GENERATION CIRCUITS

INVENTOR: Xiujun Guan
7551 Kirwin Lane
Cupertino, CA 95014
United States Citizen

ASSIGNEE: Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054

MARTINE & PENILLA, LLP
710 Lakeway Drive, Suite 170
Sunnyvale, CA 94085
Telephone (408) 749-6900

PROCESSOR CORE CLOCK GENERATION CIRCUITS

by Inventor

Xiujun Guan

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] This invention relates generally to clock generation and more specifically, to internal custom clock frequency generation.

2. Description of the Related Art

[0002] Designs of advanced superscalar microprocessor architectures require reliable timing relationships between the various frequencies propagated within the superscalar microprocessor architecture. An example of one superscalar microprocessor architecture is shown in Figure 1, which is a diagram of a computer 100 illustrating a central processing unit (CPU) 110 with a microcore (core) 120. Computer 100 can have multiple CPUs 110 connected to one or more memory 160 elements via a system bus 170. Typically, memory 160 compositions include DRAM, SRAM or flip-flops, which function as storage for data and instructions. Within CPU 110, one or more cores 120 use an interconnect 150 to transfer data and instructions to a cache 130. Further, a control logic 140 uses interconnect 150 to control the flow of data and instructions within CPU 110.

[0003] Typically, current CPUs 110 operate at higher frequencies than other motherboard components, such as hard-wired device drivers and memory 160. Consequently, motherboard designs incorporated methods to manipulate the different frequencies to enable proper motherboard operation. Similarly, internal elements of

cores 110 operate at multiple frequencies. Current optimal superscalar microprocessor architectures insert delay circuitry, which forces faster, higher frequency component to wait, while slower, lower frequency components process data and instructions.

[0004] Figure 2 is a diagram illustrating elements of core 120 (FIG. 1) in CPU 110. Core 120 can include an instruction cache 210, an instruction fetch unit (IFU) 220, multiple integer execution units (IEU) 230 and multiple floating-point graphics units (FGUs) 240. An FGU interconnect 250 connects the output from FGU 240 to IFU 220. Typically, IFU 220 retrieves data and instructions from instruction cache 210. If necessary, core 120 can also retrieve data and instructions from cache 130 and memory 160. Each IEU 230 includes an arithmetic logic unit (ALU) for computation in addition to other logic elements. Further, each IEU 230 connects to one FGU 240. Within each FGU 240 is a multiplier pipeline and an adder pipeline, which perform floating-point arithmetic and other graphics computations. Ultimately, output from FGU 240 travels via FGU interconnect 250 for use by IFU 220.

[0005] One problem with the design illustrated in Figure 2 is the one-to-one relationship between IEU 230 and FGU 240. Because each FGU 240 has a multiplier pipeline and an adder pipeline, core 120 uses eight pipelines. As the number of IEUs 230 increase to exploit parallel computation, the number of FGUs 240 correspondingly increase. This results in increased circuitry in core 120 and CPU 110. A possible solution to the problem of increasing circuitry is to remove elements within core 120. However, while this solution reduces circuitry, another problem results.

[0006] Each IEU 230 synchronizes operations to the CPU system clock. In the example shown on Figure 2, during one clock cycle, each IEU 230 sends output to an

FGU 240. Upon removal of an FGU 240, only three FGUs 240 remain to process four IEU 230 outputs. Similarly, upon removal of another FGU 240, only two FGUs 240 remain to process four IEU 230 outputs. This requires the remaining FGUs 240 to delay the IEU 230 outputs to handle each output separately.

[0007] Accordingly, what is needed is a solution to reduce circuitry on a core 120 while adhering to the goals of designing optimal superscalar microprocessor architectures.

SUMMARY OF THE INVENTION

[0008] Broadly speaking, the present invention fills these needs by providing one embodiment of a processor defined by a cache capable of storing data, a control logic capable of controlling a flow of data and at least one core coupled to the cache and the control logic, capable of generating a multiple of a CPU clock signal. The core is further defined by an instruction cache capable of storing data, an instruction fetch unit capable of fetching data, a plurality of integer execution units coupled to the instruction fetch unit and a single floating point graphics unit coupled to the plurality of integer execution units including circuitry capable of generating the multiple of the CPU clock signal.

[0009] One embodiment of the invention is also defined by a circuit with a floating point graphics unit having a voltage control delay line unit capable of generating multiple delayed clock signals, a phase frequency detector coupled to the voltage control delay line unit capable of detecting phase differences, and a charge pump coupled to the phase frequency detector and the voltage control delay line unit capable of increasing or decreasing voltage. The circuit is further defined by at least one symmetric NOR coupled to the voltage control delay line unit capable of combining signals with identical rising edges, at least one symmetric NAND coupled to at least one symmetric NOR capable of combining signals with identical falling edges and a buffer coupled to at least one symmetric NAND capable of buffering the multiple of the CPU clock signal.

[00010] A method for generating a custom clock frequency is also disclosed. The method includes receiving a CPU clock signal, delaying the CPU clock signal with at least two inverters and generating a plurality of output signals. The method also

includes combining a plurality of signals from the plurality of output signals to generate a combined custom clock signal.

[00011] Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[00012] The invention, together with further advantages thereof, may best be understood by reference to the following description taken in conjunction with the accompanying drawings in which:

[00013] Figure 1 is a diagram of a computer illustrating a central processing unit (CPU) with a microcore (core);

[00014] Figure 2 is a diagram illustrating elements of a core in the CPU;

[00015] Figure 3 is a diagram illustrating the elements of a core in the CPU, in accordance with an embodiment of the present invention;

[00016] Figure 4 is a high-level diagram illustrating the generation of a multiple of a CPU clock signal using a multiplier pipeline, in accordance with an embodiment of the present invention;

[00017] Figure 5 is a circuit diagram illustrating circuit elements, in accordance with an embodiment of the present invention;

[00018] Figure 6 is a circuit diagram illustrating circuit elements within a voltage control delay line unit, in accordance with an embodiment of the present invention;

[00019] Figure 7A is a circuit diagram illustrating circuit elements within a charge pump, in accordance with an embodiment of the present invention;

[00020] Figure 7B is a graph illustrating a signal lock of a control signal, in accordance with an embodiment of the present invention;

[00021] Figure 8 is a circuit diagram illustrating circuit elements within a symmetric NOR, in accordance with an embodiment of the present invention;

[00022] Figure 9 is a circuit diagram illustrating circuit elements within a symmetric NAND, in accordance with an embodiment of the present invention;

[00023] Figure 10 is a timing diagram illustrating the generation of the multiple of the CPU clock signal, in accordance with an embodiment of the present invention; and

[00024] Figure 11 is a flowchart showing a method for generating the multiple of the CPU clock signal, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[00025] An invention is disclosed for circuitry and methods for generating a custom clock frequency in a central processing unit (CPU). For example, in one embodiment of the invention, circuitry reduction on a microcore (core) can occur by merging multiple circuit elements, such as merging multiple floating-point graphics units (FGUs) into one floating-point graphics unit (FGU). During one CPU clock cycle, the FGU processes multiple IEU input signals. Thus, the FGU operates at a higher frequency than the CPU system clock to process the IEU input signals. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent however, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order not to unnecessarily obscure the present invention.

[00026] Figure 3 is a diagram illustrating the elements of a core 320 in CPU 110 (FIG. 1), in accordance with an embodiment of the present invention. Core 320 receives multiple FGU input 335 signals from four IEUs 230 (FIG. 2). After FGU 340 completes floating point and graphics computations, output travels to IFU 220 via a bi-directional FGU output 350. Core 320 differs from core 120 by the merging of four FGUs 240 into one FGU 340. Thus, FGU 340 is capable of simultaneously processing four IEU 230 outputs via FGU inputs 335. To enable FGU 340 computations, FGU 340 includes one multiplier pipeline and one adder pipeline. Figure 4 is a high-level diagram 400 illustrating the generation of a multiple of a CPU clock signal using a multiplier pipeline, in accordance with an embodiment of the present invention.

[00027] In Figure 4, IEU 230 (FIG. 2) signals enter FGU 340 (FIG. 3) via FGU inputs 335. A latch 410 receives the IEU 230 input signal, which propagates to a merge unit 420. Within merge unit 420, four IEU 230 input signals generated during one CPU clock cycle merge into a higher frequency signal operating at four times the frequency of the CPU system clock. The higher frequency signal is then propagated to a multiplier pipeline 430. Although an adder pipeline is not shown, the adder pipeline has a similar structure to multiplier pipeline 430. After multiplication and addition operations, the higher frequency signal propagates through a demultiplexer 440 to send output to IFU 220 via FGU output 350. It should be apparent that other configurations of the circuit elements in high-level diagram 400 are possible without departing from the purpose of generating a custom clock frequency. For example, any number of IEUs 230 can generate input signals to merge unit 420.

[00028] Figure 5 is a circuit diagram illustrating circuit elements, in accordance with an embodiment of the present invention. In one exemplary embodiment, within merge unit 420 (FIG. 4) is a delay-locked loop (DLL) 500 circuit capable of generating a custom clock frequency. A CPU clock signal 515 enters the DLL 500 circuit and propagates to a voltage control delay line unit (VCDL) 510. A control signal 525, an internal signal of the DLL 500 circuit, also enters VCDL 510. More circuitry within VCDL 510, as shown in Figure 6, further generate multiple output signals. The multiple output signals, or multiple delayed clock signals, can be a first signal 535, a last signal 545, a first output signal 532, a second output signal 534, a third output signal 536 and a fourth output signal 538. First signal 535 and last signal 545 propagate to a phase frequency detector (PFD) 520, which detects a phase difference between first signal 535 and last signal 545. PFD 520 then generates a

down voltage signal 555 and an up voltage signal 565, both of which enter a charge pump 530. Thereafter, charge pump 530 adjusts control signal 525.

[00029] The remaining output signals from VCDL 510 enter two symmetric NORs 540. First output signal 532 and third output signal 536 enter one symmetric NOR 540 while second output signal 534 and fourth output signal 538 enter another symmetric NOR 540. The symmetric NORs 540 then generate a first combined output signal 542 and a second combined output signal 544. These two signals enter a symmetric NAND 550, which generates a custom combined clock signal 560. Custom combined clock signal 560 is then buffered with a buffer 570 for output to the multiplier and adder pipelines as shown in Figure 4.

[00030] A phase-locked loop (PLL) circuit can generate a custom clock frequency. Specifically, the PLL circuit can generate a clock signal outside core 320 (FIG. 3). However, this does not solve the problem of generating a higher frequency from within core 320 based on CPU clock signal 515. Alternatively, the PLL circuit can generate a clock signal within core 320. However, this solution suffers from jitter, or noise, because of the higher frequencies generated in core 320.

[00031] Further, a conventional DLL circuit can generate a custom clock frequency. However, conventional DLL circuits suffer from false locks and harmonic locks. False locks are the locking of signals at improper frequencies and harmonic locks occur via a signal lock to a non-deterministic multiple of CPU clock signal 515. In place of the PLL circuit and the conventional DLL circuit, an embodiment of the invention includes using the DLL 500 circuit to generate a custom clock frequency. As is later shown in Figure 7A, one exemplary embodiment resolves problems with jitter, false locks and harmonic locks.

[00032] Figure 6 is a circuit diagram illustrating circuit elements within VCDL 510 (FIG. 5), in accordance with an embodiment of the present invention. CPU clock signal 515 propagate through a delay unit 610 and a pulse generator 620. In one exemplary embodiment, delay unit 610 includes eight inverters. Each inverter delays CPU clock signal 515 by one-eighth of one CPU clock cycle. The delay occurs because control signal 525, which is precisely locked to an analog value, is used by the inverters to produce a one-eighth delay of CPU signal 515. The mechanism for a precise signal lock is later illustrated in Figure 7B.

[00033] Accordingly, each inverter pair delays CPU clock signal 515 by one-quarter of one CPU clock cycle. Further, control signal 525 controls the signals delayed by each inverter. For example, when CPU clock signal 515 enters delay unit 610, CPU clock signal 515 uses a first inverter path 650 to enter a first inverter 630. Then, CPU clock signal 515 propagates through all inverters to a last inverter 640. While CPU clock signal 515 propagates through the inverters, control signal 525 controls the delay of the same inverters and by using pulse generator 620, generates first output signal 532, second output signal 534, third output signal 536 and fourth output signal 538. Specifically, pulse generator 620 generates a signal by using a rising edge of CPU clock signal 515 through one inverter and a falling edge of CPU clock signal 515 through a second inverter. Thus, as an example, pulse generator 620 uses the rising edge of the signal propagated through first inverter path 650 and the falling edge of the signal propagated through an inverter path 660 to generate first output signal 532.

[00034] Returning to Figure 5, VCDL 510 generates multiple output signals that propagate to PFD 520 or symmetric NORs 540. After PFD 520 generates down

voltage signal 555 and up voltage signal 565, charge pump 530 locks in a voltage value to generate control signal 525. Figure 7A shows a circuit diagram illustrating circuit elements such as a Schmitt circuit 710, within charge pump 530, in accordance with an embodiment of the present invention. Control signal 525 is an analog value, which sets the delays for the inverters in delay unit 610 (FIG. 6). If delay unit 610 is too slow, then charge pump 530 increases the voltage. Alternatively, if delay unit 610 is too fast, then charge pump 530 decreases the voltage.

[00035] In one embodiment, charge pump 530 (FIG. 5) includes Schmitt circuit 710 (FIG. 7A) to avoid and prevent false locks and harmonic locks. Figure 7B is a graph 700 illustrating a signal lock of control signal 525, in accordance with an embodiment of the present invention. Graph 700 shows control signal 525 voltage values along an x-axis and clock speed values along a y-axis. For example, if CPU clock signal 515 can equal any value from about 0MHz to about 400MHz, then a multiple of CPU clock signal 515 has a voltage value between about 0.3V 760 and about 0.8V 780. It should be apparent that the voltage values and clock speeds are exemplary and that many other values are possible within any defined voltage value range, as long as charge pump 530 increases and decreases control signal 525 to an appropriate voltage level to enable a precise signal lock. For example, if control signal 525 is above approximately 0.8V, then Schmitt circuit 710 decreases the voltage. Thus, by manipulating control signal 525, charge pump 530 avoids and prevents false locks and harmonic locks.

[00036] VCDL 510 also generates output signals to symmetric NORs 540. Figure 8 is a circuit diagram illustrating circuit elements within a symmetric NOR 540, in accordance with an embodiment of the present invention. Typically, a standard NOR

gate (not shown) does not properly balance rising edges of input signals when the input signals have high frequencies. Accordingly, when the NOR gate receives two signals, the rising edge of one signal will not be substantially identical to the rising edge of the second signal. In contrast, symmetric NORs 540 receiving two signals such as first signal 810 and a second signal 820 have substantially identical rising edges. Thus, symmetric NORs 540 can generate first combined output signal 542 and second combined output signal 544.

[00037] Figure 9 is a circuit diagram illustrating circuit elements within a symmetric NAND 550 (FIG. 5), in accordance with an embodiment of the present invention. Symmetric NAND 550 (FIG. 5) generates custom combined clock signal 560 by providing substantially identical falling edges for first combined output signal 542 and second combined output signal 544. Similar to the problem encountered by using a standard NOR gate, a standard NAND gate (not shown) cannot provide balanced falling edges.

[00038] In another embodiment, two symmetric NANDs 550 and one symmetric NOR 540 can replace two symmetric NORs 540 and one symmetric NAND 550, respectively. Accordingly, inputs to the symmetric NANDs will have substantially identical falling edges to generate first combined output signal 542 and second combined output signal 544. Then, symmetric NOR 540 will generate custom combined clock signal 560 by using substantially identical rising edges of the signals.

[00039] Figure 10 is a timing diagram 1000 illustrating the generation of the multiple of the CPU clock signal, in accordance with an embodiment of the present invention. A CPU clock cycle 1060 is one clock cycle of CPU clock signal 515. Thus, a first timing signal 1010 corresponds to first output signal 532 and a second

timing signal 1020 corresponds to second output signal 534. Similarly, a third timing signal 1030 corresponds to third output signal 536 and a fourth timing signal 1040 corresponds to fourth output signal 538. When symmetric NOR 540 combines first output signal 532 and third output signal, first timing signal 1010 and third timing signal 1030 combine to form first combined output signal 542. Similarly, when symmetric NOR 540 combines second output signal 534 and fourth output signal 538, second timing signal 1020 and fourth timing signal 1040 combine to form second combined output signal 542. Consequently, after symmetric NAND 550 generates custom combined clock signal 560, a combined timing signal 1050 shows a higher frequency clock operating at four times the speed of CPU clock cycle 1060. Thus, the higher frequency clock is the multiple of the CPU clock. Although eight inverters produce a high frequency clock speed operating at four times the CPU clock speed, more or less inverters can be used to generate other custom clock frequencies.

[00040] Figure 11 is a flowchart 1100 showing a method for generating the multiple of the CPU clock signal, in accordance with an embodiment of the present invention. The method begins when VCDL 510 (FIG. 5) receives a clock signal in step 1110. Then, VCDL 510 receives control signal 525 from charge pump 530 in step 1120. In the following step 1130, delay unit 610 (FIG. 6) delays the CPU clock signal and in step 1140, pulse generator 620 outputs signals. In step 1150, if the output signals are first and last signals from first inverter 630 and last inverter 640, respectively, then the method proceeds to step 1160, where PFD 520 and charge pump 530 regulate control signal 525. Alternatively, if the output signals are not first and last signals from first inverter 630 and last inverter 640, then the method proceeds to step 1170. In this step, symmetric NORs 540 and symmetric NAND 550 combine

signals to generate custom combined clock signal 560. Thereafter, the method ends and repeats during another CPU clock cycle 1060 (FIG. 10).

[00041] Embodiments of the present invention may be practiced with various computer system configurations including hand-held devices, microprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers and the like. The invention can also be practiced in distributing computing environments where tasks are performed by remote processing devices that are linked through a wire-based or wireless network.

[00042] With the above embodiments in mind, it should be understood that the invention can employ various computer-implemented operations involving data stored in computer systems. These operations are those requiring physical manipulation of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared and otherwise manipulated.

[00043] Any of the operations described herein that form part of the invention are useful machine operations. The invention also relates to a device or an apparatus for performing these operations. The apparatus can be specially constructed for the required purpose, or the apparatus can be a general-purpose computer selectively activated or configured by a computer program stored in the computer. In particular, various general-purpose machines can be used with computer programs written in accordance with the teachings herein, or it may be more convenient to construct a more specialized apparatus to perform the required operations.

[00044] The invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage

device that can store data, which can be thereafter be read by a computer system. Examples of the computer readable medium include hard drives, network attached storage (NAS), read-only memory, random-access memory, CD-ROMs, CD-Rs, CD-RWs, magnetic tapes and other optical and non-optical data storage devices. The computer readable medium can also be distributed over a network-coupled computer system so that the computer readable code is stored and executed in a distributed fashion.

[00045] Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications can be practiced within the scope of the appended claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

What is claimed is: